Modulated Convolutional Networks

Zhang, Baochang; Wang, Runqi; Wang, Xiaodi; Han, Jungong; Ji, Rongrong

Published in:
IEEE Transactions on Neural Networks and Learning Systems

DOI:
10.1109/TNNLS.2021.3060830

Publication date:
2021

Citation for published version (APA):
Zhang, B., Wang, R., Wang, X., Han, J., & Ji, R. (2021). Modulated Convolutional Networks. *IEEE Transactions on Neural Networks and Learning Systems*. https://doi.org/10.1109/TNNLS.2021.3060830

Modulated Convolutional Networks

Baochang Zhang, Runqi Wang, Xiaodi Wang, Jungong Han, and Rongrong Ji

***Abstract*— While the deep convolutional neural network (DCNN) has achieved overwhelming success in various vision tasks, its heavy computational and storage overhead hinders the practical use of resource-constrained devices. Recently, compressing DCNN models has attracted increasing attention, where binarization-based schemes have generated great research popularity due to their high compression rate. In this article, we propose modulated convolutional networks (MCNs) to obtain binarized DCNNs with high performance. We lead a new architecture in MCNs to efficiently fuse the multiple features and achieve a similar performance as the full-precision model. The calculation of MCNs is theoretically reformulated as a discrete optimization problem to build binarized DCNNs, for the first time, which jointly consider the filter loss, center loss, and softmax loss in a unified framework. Our MCNs are generic and can decompose full-precision filters in DCNNs, e.g., conventional DCNNs, VGG, AlexNet, ResNets, or Wide-ResNets, into a compact set of binarized filters which are optimized based on a projection function and a new updated rule during the backpropagation. Moreover, we propose modulation filters (M-Filters) to recover filters from binarized ones, which lead to a specific architecture to calculate the network model. Our proposed MCNs substantially reduce the storage cost of convolutional filters by a factor of 32 with a comparable performance to the full-precision counterparts, achieving much better performance than other state-of-the-art binarized models.**

***Index Terms*— Binarized filters, deep convolutional neural network (DCNN), discrete optimization, modulated convolutional networks (MCNs).**

I. Introduction

DEEP convolutional neural networks (DCNNs or CNNs) have attracted much attention due to their capability of learning powerful feature representations directly from raw pixels, thereby facilitating many computer vision tasks [1]–[4]. However, their success has come at the cost of having a significant amount of model parameters and expensive model training. For instance, the sizes of most DCNN models for vision applications are easily beyond hundreds of megabytes, which restricts their practical usage in most embedded platforms. This article focuses on efficient model storage, which is the key issue to save bandwidth and power consumption. Fundamentally, the huge model size is attributed to the way of filter design [5], [6], which generates a significant amount of redundant parameters that can be pruned or compressed.

With respect to DCNNs compression, one promising method is to binarize convolution filters, where the CNNs model compression is realized by using binary weights to approximate the floating-point weights [7]–[9]. Very recently, inspired by the well-known local binary pattern, local binary convolution layers are introduced in [10] that approximate the nonlinearly activated response of a standard convolutional layer. Courbariaux *et al.* [9] proposed the BinaryConnect scheme that uses the real-valued weights as a key reference for the binarization process. Later by deploying over BinaryConnect, BinaryNet [9] is introduced to train CNNs with binary weights, where activations are triggered at running time while parameters are computed at the training time. In [7], XNOR-network is presented where both the weights and inputs attached to the convolution are approximated with binary values, which allow an efficient implementation of convolutional operations, i.e., particularly by reconstructing unbinarized filters with a single-scaling factor and a binary filter. It has been theoretically and quantitatively demonstrated that simplifying the convolution procedure via binarized filters and/or approximating the original unbinarized filters is a very promising solution toward CNNs' compression.

However, the performance of binarized models generally drops significantly, compared with using the original filters. It is mainly due to the following reasons: 1) the binarization of CNNs could be essentially solved based on the discrete optimization, which, however, has long been neglected in previous works. 2) Existing methods fail to consider the quantization loss, filter loss, and intraclass compactness in the same backpropagation pipeline. 3) Rather than a single-binarized filter in use, a set of binarized filters can better approximate the full-precision convolution.

In this article, we propose a novel binarization architecture to tackle these challenges toward the highly accurate yet robust compression of CNNs. Unlike the existing work that uses a single-scaling factor in compression [7], we introduce modulation filters (M-Filters) into CNNs in order to better approximate the convolutional filters. The proposed M-Filters can help the network fuse the feature in a unified framework, which can significantly improve the network performance. To this end, a simple and specific modulation process is designed, which is replicable at each layer and can be easily implemented. A complex modulation is also bounded as in [11]. The binarized or quantization process is defined as a projection, which leads to a new loss function that can be solved within the same pipeline of the backpropagation.

Manuscript received June 20, 2020; revised October 11, 2020; accepted February 17, 2021. This work was supported in part by the Natural Science Foundation of China under Grant 62076016 and in part by the Key Research and Development Program of Shandong Province to Dianmin Sun under Grant 2019JZZY011101. The work of Baochang Zhang was supported by the Shenzhen Science and Technology Program under Grant KQTD2016112515134654. *(Baochang Zhang and Runqi Wang are co-first authors.) (Corresponding author: Baochang Zhang.)*

Baochang Zhang is with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China, and also with the Shenzhen Academy of Aerospace Technology, Shenzhen 518057, China (e-mail: bczhang@buaa.edu.cn).

Runqi Wang and Xiaodi Wang are with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China.

Jungong Han is with the Department of Computer Science, Prifysgol Aberystwyth University, Aberystwyth SY23 3DZ, U.K.

Rongrong Ji is with the School of Informatics, Xiamen University, Xiamen 361005, China.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3060830.

Digital Object Identifier 10.1109/TNNLS.2021.3060830

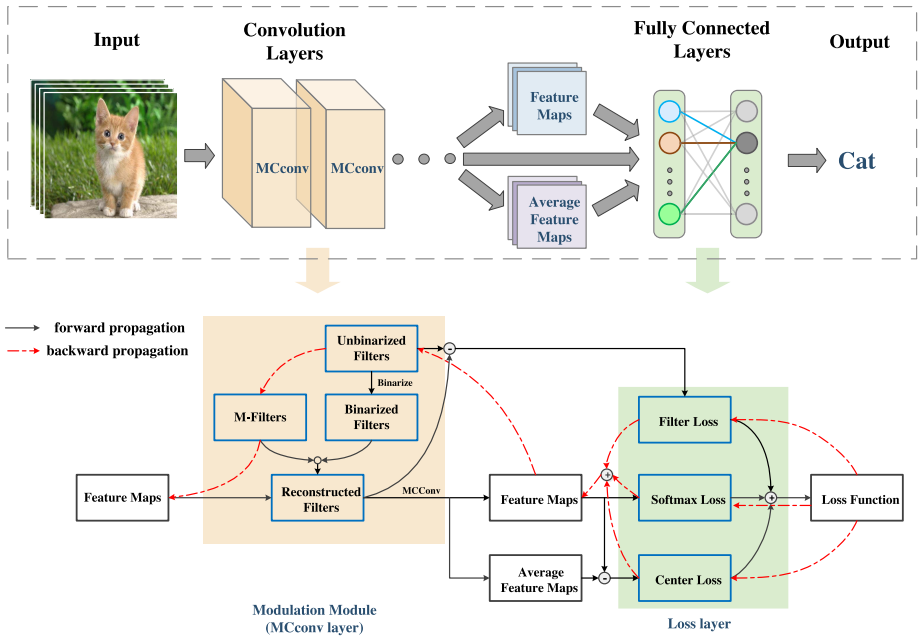


Fig. 1. MCNs are designed based on binarized convolutional filters and M-Filters. M-Filters are particularly designed to approximate unbinarized convolutional filters in the end-to-end framework. Since the operation of an M-Filter (matrix) can be shared at each layer, the model size of MCNs is marginally increased. In particular, to alleviate the disturbance caused by the binarized process, a center loss is designed to incorporate the intraclass compactness with the quantization loss and filter loss. Most importantly, our MCNs is a highly compressed model and achieves a comparable performance to the well-known full-precision Resnets and WideResnets.

In addition, we further consider the intraclass compactness in the loss function and obtain modulated convolutional networks (MCNs).[1] As shown in Fig. 1, both M-Filters and binarized filters can be jointly optimized in an end-to-end manner, resulting in a compact and portable learning architecture. Thanks to the low model complexity, such an architecture is less prone to be overfitting and is suitable for resource-constrained environments. To be specific, our MCNs reduce the required storage space of a full-precision model by a factor of 32, while achieving the best performance so far, as compared with the existing binarized filters-based CNNs, even approximating full-precision filters. MCNs are first proposed in our CVPR 2018 article [12]. In this article, we extend it in three aspects: 1) we reformulate the learning of MCNs as a discrete optimization problem and provide theoretical analysis which guarantees an optimized binarization of CNNs. The theoretical analysis was not included in [12] due to the page limit. 2) A new variant of MCNs is proposed to further improve the efficiency of MCNs, which makes our MCNs more suitable for practical applications. 3) We provide a more comprehensive evaluation of both object classification and detection tasks. More experiments on full ImageNet and object detection benchmark are added. In summary, the contributions of this article are as follows.

1) We propose to build binarized CNNs via a discrete optimization method, which can learn an optimal set of binarized filters based on a projection function and a new updated rule, in an end-to-end framework.
2) Our discrete optimization method provides a comprehensive way to calculate binarized CNNs, by considering filter loss, softmax loss, and center loss in a unified framework. We further develop M-Filters to reconstruct unbinarized filters, which leads to a specific architecture to calculate diverse CNNs. Our architecture can fuse multiple features in a unified framework.
3) The highly compressed MCNs model outperforms state-of-the-art binarized models and is comparable to full-precision counterparts.

The rest of this article is organized as follows. The related work is briefly introduced in Section II. The details of MCNs are elaborated in Section III. The implementation and experiments are described in Section IV, and conclusions are drawn in Section V.

II. Related Work

Our MCNs aim to compress the CNNs model via removing the redundancy of weights in the trained CNNs, which is in line with many exiting works. Depending on the ways of compressing CNNs weights, the prior arts can be categorized into network pruning [13]–[15], model quantization [16]–[20], sparse connections [21]–[23], and designing new CNN architectures [24]–[27]. CNNs are both computationally intensive and memory intensive, making them difficult to deploy on embedded systems. Li *et al.* [13], Li *et al.* [15], and Zhou *et al.* [16] prune the unimportant connections and retrain the network to fine-tune the weights of the remaining connections to compress the CNNs models. Li *et al.* [15] presents a simple yet efficient evaluation method based on an adaptive batch normalization, which reveals a strong correlation between different pruned CNN structures and their final accuracy. The codebook-based quantization is combined with network pruning [13] and Huffman encoding in [18]. In [28], the HashedNet method uses hash functions to reduce the model size with the SGD-based fine-tuning to improve the performance of the compressed network. Another line of

[1]The work has been commercialized.

research realizes the goal of compressing CNNs by binarizing weights and activations in deep neural networks [7]–[10]. In [19], they theoretically study how the combination of both weight and gradient quantization affects convergence. The analysis shows that weight-quantized models converge to an error related to the weight quantization resolution and weight dimension, and quantizing gradients slows convergence by a factor related to the gradient quantization resolution and dimension. As a representative, XNOR-Net approximates the convolution operation using primarily binary operations, which reconstruct unbinarized filters using binary filters with a single-scaling factor.

From the concept perspective, our idea is similar to those binarization methods that all attempt to simplify convolution procedures via binarized filters. However, the ways of approximating unbinarized filters are different. Compared with XNOR where unbinarized filters are reconstructed using binary filters with a single-scaling factor [7], ours accomplishes it by a set of M-Filters together with a set of binary filters based on a theoretical framework. On the one hand, M-Filters estimate the original convolutional filters by minimizing the filter loss using a set of optimal binary filters. The proposed M-Filters can be degenerated and simplified to single factors, which are still more effective than state-of-the-art binarized models. On the other hand, M-Filters can allow us to consider the diversity among CNNs during the optimization procedure, which can further improve the performance of compressed models.

The sparse connections have been studied to reduce the model redundancy. Toward reducing this superfluous computation, Xie *et al.* [23] propose to compute features only at sparsely sampled locations, which are probabilistically chosen according to activation responses, and then densely reconstruct the feature map with an efficient interpolation procedure. Besides, low-rank compression of filters in CNNs has been studied in [21] and [22]. The work in [24] and [25] explored model compression with specific CNN architectures, e.g., replacing regular filters with 1×1 filters. More schemes on sparse connections in CNNs can also be found in [29]–[31]. It has been theoretically shown that a sparsely connected network can achieve a certain asymptotic statistical optimality. MobileNets [24] are a class of efficient models for mobile and embedded vision applications. In MobileNets, a streamlined architecture based on depthwise separable convolutions is used to build lightweight deep neural networks. Two simple global hyperparameters are introduced that efficient tradeoff between latency and accuracy. These hyperparameters allow the model builder to choose the right sized model for their application based on the constraints of the problem. For another instance, SqueezeNet [26] is a small CNN architecture, which achieves AlexNet-level accuracy on ImageNet with $50\times$ fewer parameters.

We note that the binarization (used in MCNs) is a different way to compress the model from others, e.g., compressing deep models by reducing parameter numbers. In principle, our MCNs can be used on any full-precision CNNs, which has been tested on conventional CNNs, VGG [32], AlexNet [33], ResNets, or Wide-ResNets (WRNs) [34].

Besides, object detection is another important of computer vision, which is a resource-hungry computing task. Therefore, the research on lightweight, fast, and accurate object detection model has important academic significance and high commercial value. In the method of model compression, model quantization can greatly reduce the computation and storage capacity of the network. Li *et al.* [35] demonstrates that many of these difficulties arise because of instability during the fine-tuning stage of the quantization process and proposes several novel techniques to overcome these instabilities. They apply techniques to produce fully quantized 4-bit detectors based on RetinaNet and Faster R-CNN [1], achieving state-of-the-art performance for quantized detectors. Gao *et al.* [36] proposes DupNet which consists of two parts. First, they employ weights with duplicated channels for the weight-intensive layers to reduce the model size. Second, for the quantization-sensitive layers where quantization causes notable accuracy decline, they duplicate its input feature maps. Doing so allows using more weight channels for convolving more representative outputs. However, the research on binary object detection is not well studied. When the binarization method of the classification network is applied directly, the accuracy of the object detection model suffers. Peng and Chen [37] proposed a low bit-width weight optimization approach to train binarized neural networks (BNNs) for object detection using binary weights in both training and testing. They introduce a greedy layerwise method to train the detection network. Nevertheless, the research on the binary network is still in its infancy, where binarizing object detection networks while preserving sufficiently high accuracy is challenging.

III. Modulated Convolutional Networks

In MCNs, M-Filters are particularly designed to approximate unbinarized convolutional filters in the end-to-end framework. Each layer shares only one M-Filter, which leads to a significant reduction of the model. To alleviate the disturbance caused by the binarized process, the intraclass compactness based on the center loss is further deployed to enhance the performance. With the two schemes mentioned earlier, the performance drop is marginal even when the learnable network parameters are highly compressed.

A. Problem Statement in MCNs

To calculate the binarized or discrete filter, we design a quantization process by projecting the input onto a set as

$$\Omega := \{a_1, \ldots, a_U\}$$

where a_i and $i = 1, \ldots, U$ satisfying $a_1 < \cdots < a_U$, are defined as quantized values of the original filter. Then, we define the projection of $x \in \mathbb{R}$ onto the set Ω as

$$P_\Omega(x) = \begin{cases} a_1, & \text{if } x \le \dfrac{a_1 + a_2}{2} \\ \cdots & \\ a_i, & \text{if } \dfrac{a_{i-1} + a_i}{2} < x \le \dfrac{a_i + a_{i+1}}{2} \\ \cdots & \\ a_U, & \text{if } x > \dfrac{a_{U-1} + a_U}{2}. \end{cases} \tag{3.1}$$

TABLE I
BRIEF DESCRIPTION OF VARIABLES AND OPERATORS USED IN THIS ARTICLE

C: unbinarized filter	$\hat{C}$: binarized filter	M: modulation filter (M-Filter)
Q: reconstructed filter	$\vec{M}$: M-Filters across all layers	
i: filter index	j: plane index	K: number of planes for each filter
m: sample index	l: layer index	N: number of layers
g: input feature map index	h: output feature map index	

Proposition 1: Let $f(x)$ be a loss function, whose gradient exists. We resort to a discrete optimization method to minimize $f(x)$ to calculate binarized or quantized models. In the kth iteration, with the projection function in (3.1), $x^{[k]}$ is quantized to be $\hat{x}^{[k]}$ as

$$\hat{x}^{[k]} = P_{\Omega}\big(x^{[k]}\big).$$

A new updated rule in gradient descent is defined as

$$x \leftarrow x^{[k]} - \eta\delta_{\hat{x}}^{[k]}$$

which leads to a new optimization problem

$$\min_{x} \ f(x)$$
$$\text{subject to } x = \widehat{x}^{[k+1]}.$$

The resulting quantized model is optimal, in terms of the constraint $x = \widehat{x}^{[k+1]}$. That is, the quantized model is constrained to be the full-precision counterpart.

Proof: In order to obtain quantized parameters for $f(x)$, we resort to a new discrete optimization scheme. And we reformulate the minimization of $f(x)$ based on our projection function as

$$\min_{x} f\big(x|\hat{x}^{[k]} = P_{\Omega}\big(x^{[k]}\big)\big). \tag{3.2}$$

The new minimization problem in (3.2) is hard to solve, due to a discrete process imposed on x.

We solve (3.2) via the gradient descent method, e.g., within the backpropagation framework and define a new updated rule as

$$x \leftarrow x^{[k]} - \eta\delta_{\hat{x}}^{[k]} \tag{3.3}$$

where $[k+1]$ is dropped from x, $\delta_{\hat{x}}$ denotes the gradient of $f(x)$ with respect to $x = \hat{x}$, and η is the learning rate. The quantization process $\widehat{x}^{[k]} \leftarrow x^{[k]}$, i.e., $P_{\Omega}(x^{[k]})$, equals to finding the projection of $x + \eta\delta_{\hat{x}}^{[k]}$ onto $\mathbf{\Omega}$ as

$$\widehat{x}^{[k]} = \arg\min_{\widehat{x}} \Big\{\|\widehat{x} - x - \eta\delta_{\hat{x}}^{[k]}\|^2, \widehat{x} \in \Omega\Big\}. \tag{3.4}$$

Obviously, $\hat{x}^{[k]}$ is the solution for the above-mentioned problem. Therefore, by incorporating (3.4) into the minimization of $f(x)$, we obtain a solution for (3.2) with the Lagrangian method as

$$\min f(x) + \lambda\|\widehat{x}^{[k]} - x - \eta\delta_{\hat{x}}^{[k]}\|^2 \tag{3.5}$$

where λ is a parameter. In general, the following rule is widely accepted:

$$\widehat{x}^{[k+1]} = \widehat{x}^{[k]} - \eta\delta_{\hat{x}}^{[k]}$$

which is pugged back into (3.5), and we have

$$\min f(x) + \lambda\|\widehat{x}^{[k+1]} - x\|^2 \tag{3.6}$$

and it equals to the following minimization problem:

$$\min_{x} \ f(x)$$
$$\text{subject to } x = \widehat{x}^{[k+1]} \tag{3.7}$$

which proves the proposition. ■

We note that the newly added part shown in (3.5) is a quadratic function, and the convergence of the resulting new loss function is never affected in our extensive experiments.

B. Loss Function of MCNs

In order to constrain CNNs to have binarized weights, we introduce a new loss function in MCNs. Two aspects are considered: unbinarized convolutional filters are reconstructed based on binarized filters; the intraclass compactness is incorporated based on output features. In addition to Table I, we further introduce variables used in this section: C_i^l are unbinary filters of the lth convolutional layer, $l \in \{1, \ldots, N\}$; $\hat{C}_i^l$ denote binarized filters corresponding to C_i^l; M^l denotes the M-Filter shared by all C_i^l in the lth convolutional layer and M_j^l represents the jth plane of M^l; $\circ$ is a new plane-based operation (3.10), which is defined in Section III-C. We then have the first part of the loss function for minimization

$$L_M = \frac{\theta}{2}\sum_{i,l}\|C_i^l - \hat{C}_i^l \circ M^l\|^2 + \frac{\lambda}{2}\sum_{m}\|f_m\Big(\hat{C}, \vec{M}\Big) - \overline{f}\Big(\hat{C}, \vec{M}\Big)\|^2 \tag{3.8}$$

where θ and λ are hyperparameters, $\vec{M} = \{M^1, \ldots, M^N\}$ is M-Filters, and $\hat{C}$ is the binarized filter set across all layers. Operation $\circ$ defined in (3.10) is used to approximate unbinarized filters based on binarized filters and M-Filters, leading to the filter loss as the first term on the right-hand side of (3.8). The second term on the right is similar to the center loss used to evaluate the intraclass compactness, which is used to deal with the feature variation caused by the binarization process. $f_m(\hat{C}, \vec{M})$ denotes the feature map of the last convolutional layer for the mth sample, and $\overline{f}(\hat{C}, \vec{M})$ denotes the class-specific mean feature map of previous samples. We note that the center loss is successfully deployed to handle feature variations. To reduce the storage space, after training, we only keep the binarized filters and the shared M-Filters (quite small) to calculate the feature maps. We consider the conventional loss and then define a new loss function $L_{S,M} = L_S + L_M$, where L_S is the conventional loss function, e.g., softmax loss.

Again, based on the framework shown in Section III-A, we consider the quantization process in our loss $L_{S,M}$ and obtain the final minimization objective as

$$L\big(C, \hat{C}, M\big) = L_{S,M} + \frac{\theta}{2}\|\widehat{C}^{[k]} - C - \eta\delta_{\widehat{C}}^{[k]}\|^2 \tag{3.9}$$

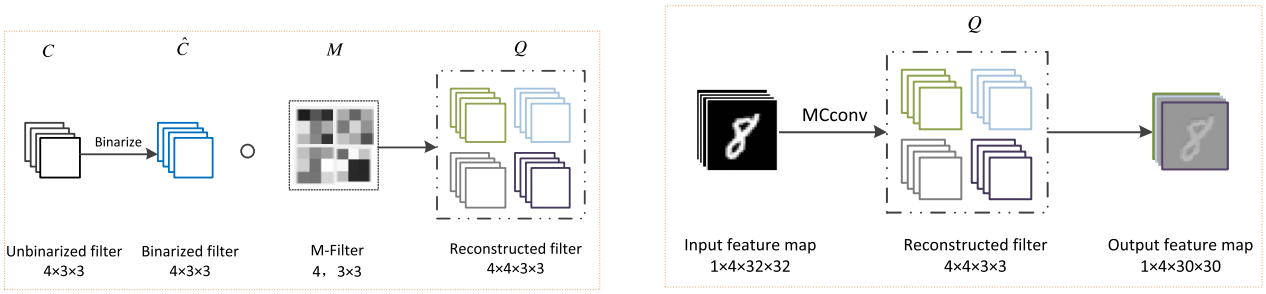


Fig. 2. (a) Modulation process based on an M-Filter to obtain a reconstructed filter Q. (b) Example of MCNs convolution with $K = 4$ planes. The number of planes of the M-Filter is the same as the number of channels of the feature map. In this article, a feature map is defined as a 3-D matrix with four channels. (a) Modulation process based on an M-Filter. (b) MCNs Convolution (MCconv).

where θ is shared with (3.8) to reduce the parameter number. $\delta_{\widehat{C}}^{[k]}$ is the gradient of $L_{S,M}$ with respect to $\widehat{C}^{[k]}$. Different from conventional method (such as XNOR) where only the filter reconstruction is considered in the weight calculation, our discrete optimization method provides a comprehensive way to calculate binarized CNNs, by considering filter loss, softmax loss, and feature compactness in a unified framework.

C. Forward Propagation With Modulation

1) Reconstructed Filters: We first design specific convolutional filters used in our MCNs. We deploy the 3-D filter across all layers with the size of $K \times W \times W$ (one filter), which has K planes, and each of the planes is a $W \times W$-sized 2-D filter. To use such kinds of filters, we extend input channels of the network, e.g., from RGB to RRRR or (RGB+X) with $K = 4$ and X denotes any channel. Note that we only use one channel of gray-level images. By doing so, we can easily implement our MCNs in existing deep learning platforms. After this extension, we directly deploy our filters in the convolution process, whose details concerning the MCNs convolution are shown in Fig. 2(b).

To reconstruct unbinarized filters, we introduce a modulated process based on M-Filters and binarized filters. An M-Filter is a matrix serving as the weight of binarized filters, which is also with the size of $K \times W \times W$. Let M_j be the jth plane of an M-Filter. We define the operation $\circ$ for a given layer as

$$\hat{C}_i \circ M = \sum_{j}^{K} \hat{C}_i * M'_j \tag{3.10}$$

where $M'_j = (M_j, \ldots, M_j)$ is a 3-D matrix built based on K copies of the 2-D matrix M_j with $j = 1, \ldots, K$. $*$ is the elementwise multiplication operator, also termed Schur product operation. In (3.10), M is a learned weight matrix, which is used to reconstruct the convolutional filters C_i based on $\hat{C}_i$ and the operation $\circ$. And it leads to the filter loss in (3.8). An example of the filter modulation is shown in Fig. 2(a). In addition, the operation $\circ$ results in a new matrix (named reconstructed filter), i.e., $\hat{C}_i * M'_j$, which is elaborated in the following. We define

$$Q_{ij} = \hat{C}_i * M'_j \tag{3.11}$$

$$Q_i = \{Q_{i1}, \ldots, Q_{iK}\}. \tag{3.12}$$

In testing, Q_i is not predefined but is calculated based on (3.11). An example is shown in Fig. 2(a). Q_i is introduced to approximate the unbinarized filters C_i in order to alleviate the information loss problem caused by the binarized process. In addition, we further require $M \geq 0$ to simplify the reconstructed process.

Since the set $\mathbf{\Omega}^n$ is separable, the projection of $C = (c_1, \ldots, c_n)^\top \in \mathbb{R}^n$ onto $\mathbf{\Omega}^n$ is calculated as

$$P_{\Omega^n}(c) = (P_\Omega(c_1), \ldots, P_\Omega(c_n))^\top.$$

That is,

$$\hat{c}_i = P_\Omega(c_i)$$

where c_i is an element of C_i, and $\hat{c}_i$ denotes the corresponding element in $\hat{C}_i$. $\hat{c}_i$ is calculated by projecting c_i onto $\mathbf{\Omega}$, whose a_i is calculated by an offline k-means clustering algorithm on the data of unbinarized filters after (per) ten epochs. Though c_i is a floating number, it can be represented as a quantization value to save the storage space.

2) Forward Propagation of MCNs Based on the MCconv Module: In MCNs, reconstructed filters Q^l in the lth layer are used to calculated output feature maps F^{l+1} as

$$F^{l+1} = \text{MCconv}\left(F^l, Q^l\right) \tag{3.13}$$

where MCconv denotes the convolution operation implemented as a new module. A simple example of forward convolutional process is described in Fig. 2(b), where there is one input feature map with one generated output feature map. In MCconv, the channels of one output feature map are generated as follows:

$$F_{h,k}^{l+1} = \sum_{i,g} F_g^l \otimes Q_{ik}^l \tag{3.14}$$

$$F_h^{l+1} = \left(F_{h,1}^{l+1}, \ldots, F_{h,K}^{l+1}\right) \tag{3.15}$$

where $\otimes$ denotes the convolution operation, and $F_{h,k}^{l+1}$ is the kth channel of the hth feature map in the $(l+1)$th convolutional layer. F_g^l denotes the gth feature map in the lth convolutional layer. In Fig. 2(b), $h = 1$ and $g = 1$, where after MCconv with one reconstructed filter, the number of the channels of the output feature map is the same as that of the input feature map.

Fig. 3 shows another example of MCNs convolution with multiple feature maps. One output feature map is the sum of

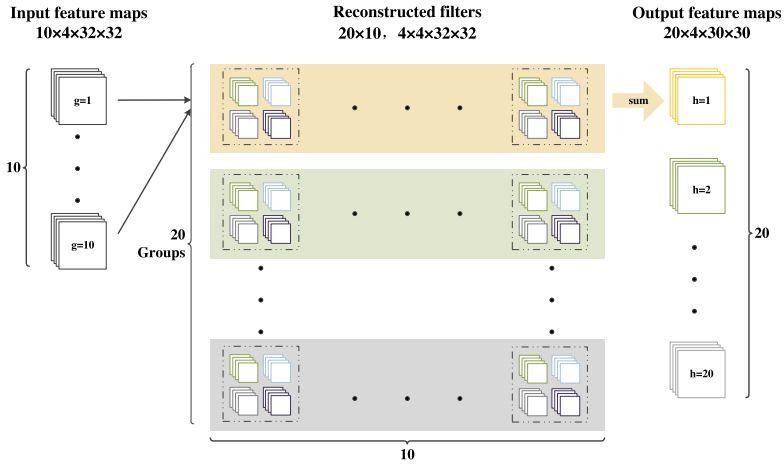


Fig. 3. MCNs Convolution (MCconv) with multiple feature maps. There are 10 and 20 feature maps in the input and the output, respectively. The reconstructed filters are divided into 20 groups and each group contains ten reconstructed filters, corresponding to the number of feature maps and MC feature maps, respectively.

the convolution between all the ten input feature maps and ten reconstructed filters in the corresponding group. For example, for the first output feature map, $h = 1, i = 1, \ldots, 10, g = 1, \ldots, 10$, and for the second output feature map, $h = 2$, $i = 11, \ldots, 20, g = 1, \ldots, 10$.

When the first convolutional layer is considered, the input size of the network is 32×32.[2] First, each channel of the image is copied $K = 4$ times, resulting in the new input of size $4 \times 32 \times 32$ to the whole network.

It is worthwhile to point out that the numbers of the input and output channels in every feature map are the same so that MCNs can be easily implemented by simply replicating the same MCconv module at each layer.

D. Backpropagation Updating

In MCNs, what needs to be learned and updated are unbinarized filters C_i and M-Filters M. These two kinds of filters are jointly learned. In each convolutional layer, MCNs sequentially update unbinarized filters and M-Filters. It should be noted that the filters are actually binarized during the forward process, which is further used to calculate the loss function. Therefore, there is no binarization process for backpropagation.

1) Updating Unbinarized Filters: We update learned filter C_i based on (3.3). $\delta_{\hat{C}}$ corresponding to C_i is defined as

$$\delta_{\hat{C}} = \frac{\partial L}{\partial \hat{C}_i} = \frac{\partial L_S}{\partial \hat{C}_i} + \frac{\partial L_M}{\partial \hat{C}_i} + \theta\left(\widehat{C}^{[k]} - C^{[k]} - \eta_1 \delta_{\widehat{C}}^{[k]}\right) \tag{3.16}$$

$$C_i \leftarrow C_i - \eta_1 \delta_{\hat{C}} \tag{3.17}$$

where L, L_S, and L_M are loss functions, and η_1 is the learning rate. Furthermore, we have

$$\frac{\partial L_S}{\partial \hat{C}_i} = \frac{\partial L_S}{\partial Q} \cdot \frac{\partial Q}{\partial \hat{C}_i} = \sum_j \frac{\partial L_S}{\partial Q_{ij}} \cdot M'_j \tag{3.18}$$

$$\frac{\partial L_M}{\partial \hat{C}_i} = \theta \sum_j \left(C_i - \hat{C}_i \circ M_j\right) \circ M_j. \tag{3.19}$$

[2]We only use one channel of gray-level images ($3 \times 32 \times 32$)

2) Updating M-Filters: We further update the M-Filter M with C fixed. δ_M is defined as the gradient of M, and we have

$$\delta_M = \frac{\partial L}{\partial M} = \frac{\partial L_S}{\partial M} + \frac{\partial L_M}{\partial M} \tag{3.20}$$

$$M \leftarrow |M - \eta_2 \delta_M| \tag{3.21}$$

where η_2 is the learning rate. Further we have:

$$\frac{\partial L_S}{\partial M} = \frac{\partial L_S}{\partial Q} \cdot \frac{\partial Q}{\partial M} = \sum_{i,j} \frac{\partial L_S}{\partial Q_{ij}} \cdot \hat{C}_i. \tag{3.22}$$

Based on (3.8), we have

$$\frac{\partial L_M}{\partial M} = -\theta \sum_{i,j} \left(C_i - \hat{C}_i \circ M_j\right) \cdot \hat{C}_i. \tag{3.23}$$

The details about derivatives with respect to the center loss can be found from [38]. The above-mentioned derivations show that MCNs are learnable with the BP algorithm. The quantization process leads to a new loss function via a simple projection function, which never affects the convergence of MCNs. We describe our algorithm in Algorithm 1.

E. MCNs-1

In our MCNs, the reconstructed filter Q_{ij} is calculated based on a matrix M'_j, in which all the elements can be potentially set to an identical value, i.e., the average of the elements in M'_j. This further reduces the size of the M matrix. If we do this for M involved in MCNs, it will end up with K groups of binarized filters, each of which is only modulated by a single factor. We name these special MCNs with a single element in each M'_j matrix as MCNs-1. Revealed in our experimental results, MCNs-1 is a compact version of MCNs without causing large performance loss.

In particular, in MCNs-1, it is not necessary to calculate Q in the forward process. Alternatively, we just produced the average of M on the output feature maps. Differently, in the backpropagation process, we still use the original updated rule for M, which can bring diversities among learned filters and convolutional features.

Algorithm 1 MCNs Training. L Is the Loss Function, Q Is the Reconstructed Filter, λ_1 and λ_2 Are Decay Factors, and N Is the Number of Layers. Binarize() Binarizes the Filters Obtained via the Projection (3.1), and Update() Updates the Parameters Based on Our Update Scheme

Input: a minibatch of inputs and their labels, unbinarized filters C, modulation filters M, learning rates η_1 and η_2, corresponding to C and M, respectively.

Output: updated unbinarized filters C^{t+1}, updated modulation filters M^{t+1}, and updated learning rates η_1^{t+1} and η_2^{t+1}.

{1. Computing gradients with aspect to the parameters:}
{1.1. Forward propagation:}
for $k = 1$ to N **do**
  $\hat{C} \leftarrow$ Binarize(C) (using Eq. 3.1)
  Computing Q via Eq. 3.11 - 3.12
  Convolutional features calculation using Eq. 3.13 - 3.15
end for
{1.2. Backward propagation:}
{Note that the gradients are not binary.}
Computing $\delta_Q = (\partial L / \partial Q)$
for $k = N$ to 1 **do**
  Computing $\delta_{\hat{C}}$ using Eq. 3.16, Eq. 3.18 - 3.19
  Computing δ_M using Eq. 3.20, Eq. 3.22 - 3.23
end for
{Accumulating the parameters gradients:}
for $k = 1$ to N **do**
  $C^{t+1} \leftarrow$ Update($\delta_{\hat{C}}$, η_1) (using Eq. 3.17)
  $M^{t+1} \leftarrow$ Update(δ_M, η_2) (using Eq. 3.21)
  $\eta_1^{t+1} \leftarrow \lambda_1 \eta_1$
  $\eta_2^{t+1} \leftarrow \lambda_2 \eta_2$
end for

IV. Implementation and Experiments

We evaluate MCNs on two tasks, i.e., object classification and detection, on six data sets, including MNIST, Street View House Numbers (SVHN), CIFAR-10, CIFAR-100, ImageNet, and PASCAL VOC 2007. It can be applied to any CNNs, including conventional CNNs, VGG-16 [32], AlexNets [33], and also ResNets [34]. In particular, our new modulation module is applied to WRNs [39]. In our experiments, four NVIDIA GeForce GTX 1080ti GPUs or four TitanX are used. In what follows, the term U-MCNs is short for unbinarized MCNs or full-precision MCNs implemented only based on L_S without the binarized process involved.

A. Data Sets and Implementation Details

1) Data Sets: The MNIST [40] data set is composed of a training set of 60 000 and a testing set of 10 000 32×32 gray-scale images of hand-written digits from 0 to 9.

CIFAR-10 [41] is a natural image classification data set containing a training set of 50 000 and a testing set of 10 000 32×32 color images across the following ten classes: airplanes, automobiles, birds, cats, deers, dogs, frogs, horses, ships, and trucks. Differently, CIFAR-100 consists of 100 classes.

The SVHN data set [42] is a real-world image data set taken from Google Street View images. It contains MNIST-like 32×32-sized images centered on a single character, which, however, includes a plethora of challenges, such as illumination changes, rotations, and complex backgrounds. The data set consists of 600 000 digit images: 73 257 digits for training, 26 032 digits for testing, and 531 131 additional images. The additional images are not used in the training of MCNs.

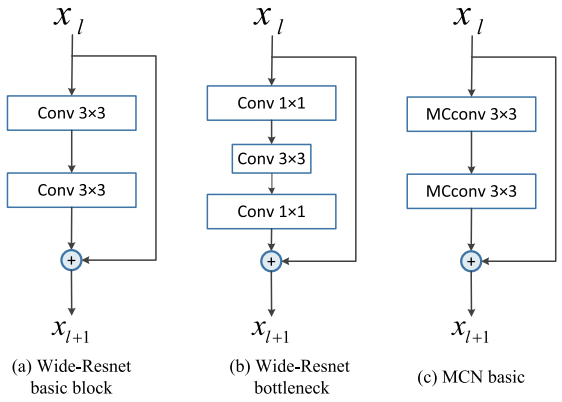


Fig. 4. Residual blocks. (a) and (b) WRNs. (c) Basic block for MCNs.

The ImageNet ILSVRC-2012 classification data set [43] consists of 1000 classes, with 1.28 million images for training and 50 000 images for validation. Different from MNIST, SVHN, and CIFAR, ImageNet consists of images with much higher resolutions. In addition, each image usually contains more than one attribute, which may have a large impact on the classification accuracy. We first follow LBCNN to use a 100-class subset of ImageNet2012 [43] to evaluate our proposed method. The 100 classes are randomly selected from the full ImageNet data set, and similar subsets are also used in [10], [44], and [45]. To further validate our methods, we also test the full ImageNet2012. The resulting model is also tested on the object detection benchmarks, such as PASCAL VOC 2012 and MS COCO.

2) Implementation Details: On all data sets, the size of each M-Filter and also convolutional filters is $4 \times 3 \times 3$ ($K = 4$). We replace spatial convolution layers with MCconv modules, as shown in Fig. 2. In all experiments, we adopt max-pooling and ReLU after convolution layers, and a dropout layer [45] after the FC layer to avoid overfitting. On CIFAR-10, CIFAR-100, SVHN, and ImageNet data sets, we evaluate MCNs based on WRNs or VGG. The basic blocks in WRNs and MCNs are shown in Fig. 4. The WRNs divide the whole network into four stages. The bottleneck structure is not used in MCNs since the 1×1 kernel does not propagate any M-filter information. The structures of both WRNs and MCNs are the same except that the Conv in WRNs is replaced by MCconv. Fig. 5 shows the details of network architectures of CNNs and MCNs. The initial values of η_1 and η_2 are set to 0.1 and 0.01, respectively. The weight decays are set to 0.2.

B. Parameters Evaluation

1) θ and λ: There are θ and λ in (3.8) which are related to the filter loss and center loss. The effect of parameters θ and λ are evaluated on CIFAR-10 for a 20-layer MCN with width

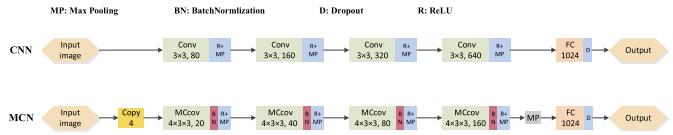


Fig. 5. Network architectures of CNNs and MCNs.

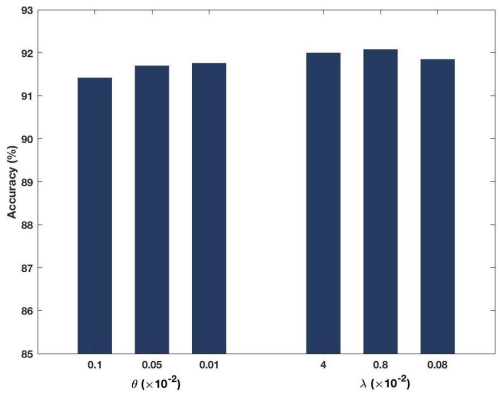


Fig. 6. Accuracy with different θ and λ.

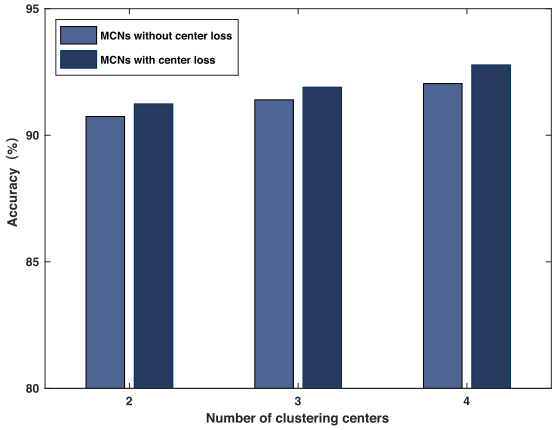


Fig. 7. Accuracy with different numbers of clustering centers for 20-layer MCNs with width 16-16-32-64.

16-16-32-64, the architectural detail of which can be found in [39]. The Adadelta optimization algorithm [46] is used during the training process, with the batch size 128. Using different values of θ, the performance of MCNs is shown in Fig. 6. First, only the effect of θ is evaluated and then the center loss is implemented based on a fine-tuning process. It is observed that the performance is stable with varying θ and λ.

2) Number of Clustering Centers: In Section III-C1, we show the quantization with $U = 2, 3, 4$ denoting numbers of clustering centers in (3.1). In this experiment, we investigate the effect on varying the number of clustering centers in MCNs based on CIFAR-10.

The results are shown in Fig. 7, where we can see that the accuracy increases with more clustering centers, and the center loss can also be used to improve the performance. However, to save storage space and to compare with other

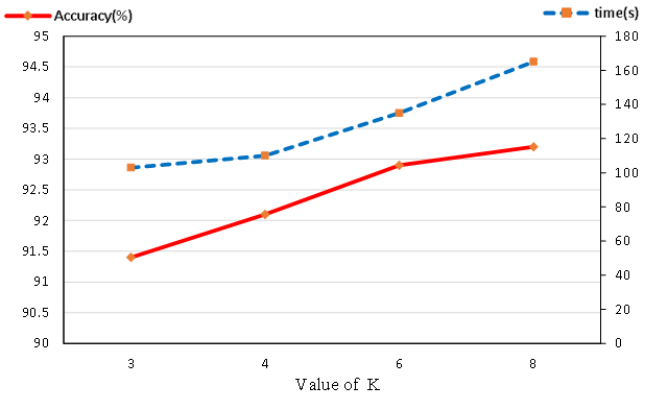


Fig. 8. Accuracy and time of train with different K for 20-layer MCNs with width 16-16-32-64 on CIFAR-10.

binary networks, we use two clustering centers for MCNs in all the following experiments.

Our binarized networks can save the storage space by a factor of 32 in convolutional layers compared with corresponding full-precision networks where 4 bytes (32 bits) are used to represent a real value. Since MCNs only contain one fully connected layer that is not binarized, the storage of the whole network is significantly reduced.

3) Architecture Parameter K: The number of planes for each M-filter, i.e., K, is also evaluated. Revealed by the results in Fig. 8, more planes in each M-filter involved in reconstructing the unbinarized filters yield better performance. For example, when increasing K from 4 to 8, the performance is improved by 1.02%, even with a 55.4% increase in the training time. For simplicity, we choose $K = 4$ in the following experiments.

4) Width of MCNs: CIFAR-10 is used to evaluate the effect of the width of WRNs with MCNs. The accuracy and number of parameters are compared with a recent binary CNN, LBCNN. The basic width of the stage (the number of convolution kernels per layer) is set to 16-16-32-64. To compare with LBCNN, we set up 20-layer MCNs with basic block-c (in Fig. 4), whose depth is the same as in LBCNN. We also use other network widths to evaluate the effect of width on MCNs.

The results are shown in Table II. The second column refers to the width of each layer of the MCNs and a similar notation is also used in [39]. In the third column, we give the parameter amounts of MCNs and the 20-layer LBCNN with the best result. The fourth column shows the accuracy of baselines whose networks are trained based on the WRNs structure with the same depth and width as MCNs. The last two columns show the accuracies of U-MCNs and MCNs, respectively. The performance in the last three columns shows that the accuracy of MCNs only decreases a little when binarized filters are used. Note that with a fixed number of convolutional layers, the performance of MCNs increases with larger network width. At the same time, the number of parameters also increases. Compared with LBCNN, the parameters of the MCNs are much fewer (61 versus 17.2 M), but the performance of MCNs

TABLE II
CLASSIFICATION ACCURACY (%) ON CIFAR-10 WITH 20-LAYER U-MCNS AND MCNS

Method	Network stage kernels	Parameter amount (M)	WRNs	U-MCNs	**MCNs**	**MCNs-1**
MCNs	16-16-32-64	1.1	92.31	93.69	92.08	92.10
	16-32-64-128	4.3	–	94.88	93.98	93.94
	32-64-128-256	17.1	–	95.50	95.13	95.33
	64-64-128-256	17.2	95.75	95.72	95.30	95.34
LBCNN (q=384)	–	61	–	–	92.96	–

TABLE III
CLASSIFICATION ACCURACY (%) ON 4 DATA SETS

Method	**MCNs-1**	**MCNs**	LBCNN [10]	BinaryConnect [9]	BNN [8]	XNOR-Net [7]	ResNet-101 [34]	Maxout [47]	MIN [48]
MNIST	–	99.52	99.51	98.99	98.60	–	–	99.55	99.53
CIFAR-10	95.47	95.39	92.99	91.73	89.85	89.83	93.57	90.65	91.19
SVHN	–	96.87	94.50	97.85	97.49	–	–	97.53	97.65
CIFAR-100	77.96	78.13	–	–	–	–	74.84	61.43	64.32

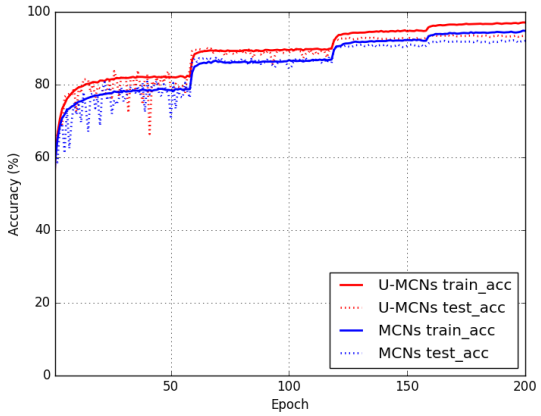


Fig. 9. Training and testing curves.

is much better (92.96% versus 95.30%). Besides, the last three columns show that MCNs have achieved similar performance to U-MCNs and WRNs.

C. Model Effect

1) Learning Convergence: The MCNs model is based on a binarized process, which is implemented on the torch platform (classification). For a 20-layer MCN with width 16-16-32-64 that is trained after 200 epochs, the training process takes about 3 h with two 1080ti GPUs. We plot the training and testing accuracy of MCNs and U-MCNs in Fig. 9. The architecture of U-MCNs is the same as that of MCNs. Fig. 9 clearly shows that MCNs (the blue curves) converge at similar speeds to their unbinarized counterpart (the red curves).

2) Runtime Analysis: We have conducted runtime analysis to compare MCNs and LBCNN. The runtimes of MCNs and LBCNN for all the test samples of CIFAR-10 are 8.7 and 160.6 s, respectively, with similar accuracy (93.98% versus 92.96%). When LBCNN has a similar amount of parameters (4.3M) to MCNs, the test runtime of LBCNN becomes 16.2 s, which is still slower than our MCNs.

3) Visualization: We visualize MCconv features in Fig. 10 across different layers and the curves of elements in different M-Filters in Fig. 11. Similar to conventional CNNs, features from different layers capture the rich and hierarchy information in Fig. 10. Based on the reconstructed filters Q corresponding to M-Filters, we obtain convolutional features, which appear to be diverse for different M-Filters. In summary, different MCconv layers and M-Filters can capture the hierarchy and diverse information, which, thus, results in a high performance based on compressed models. Fig. 11 shows the curves of the elements in M-Filter 1 (M_1'), M-Filter 2 (M_2'), M-Filter 3 (M_3'), and M-Filter 4 (M_4') [in Fig. 2(a) and (3.10)] on the CIFAR experiment. The values of nine elements in each M-Filter are learned similar to their averages (dotted lines), which validates that the special MCNs-1 with a single-average element in each M_j' matrix is reasonable and compact without the performance loss.

D. Experiments on MNIST, SVHN, and CIFAR-10/100

Table III shows image classification results from our experiments on various data sets. MCNs are compared with state-of-the-art methods, such as LBCNN [10], BinaryConnect [9], BNN [8], XNOR-net [7], ResNet-101 [34], maxout network [47], and network in network [48]. For each data set, the training methods and parameters of MCNs models are described in the following sections.

1) MNIST: Fig. 5 shows the details of network architectures of MCNs used on MNIST. Due to the easy task on MNIST, the architectures of our MCNs are based on a simple CNN. The MCNs contain four MCconv layers and one fully connected layer. For this model, we adopt max-pooling and ReLu after convolution layers, and a dropout layer [45] after the FC layer to avoid overfitting. We report the performance of our algorithm on a test set after 200 epochs on the average over five predictions. The results are shown in Table III. It is observed from the experiments that MCNs achieve 99.52% accuracy on the MNIST test, which is better than other binary methods and comparable to other full-precision models.

2) SVHN: The network depth is set to 28 and the stage is set to 64-64-128-256. The total training epochs are 200 and the learning rate is reduced per 30 epochs. The results are list

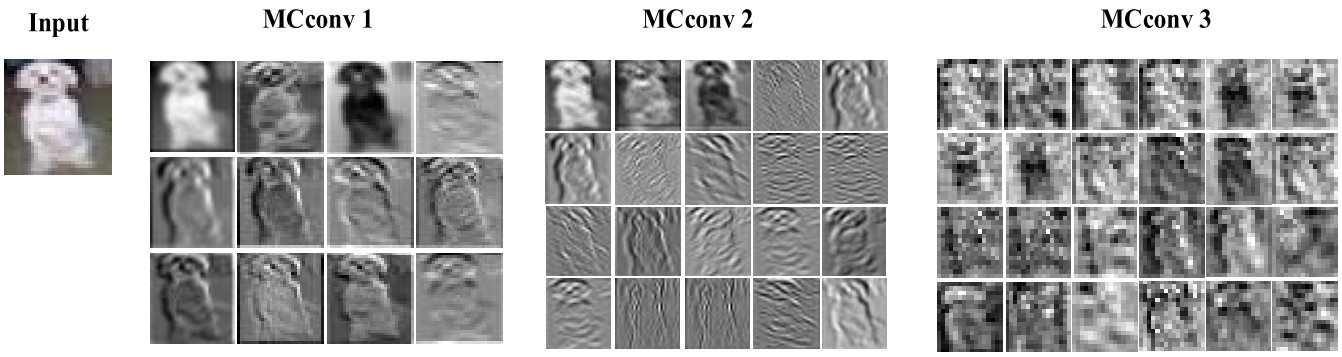


Fig. 10. Example of output feature maps produced by Q from different layers.

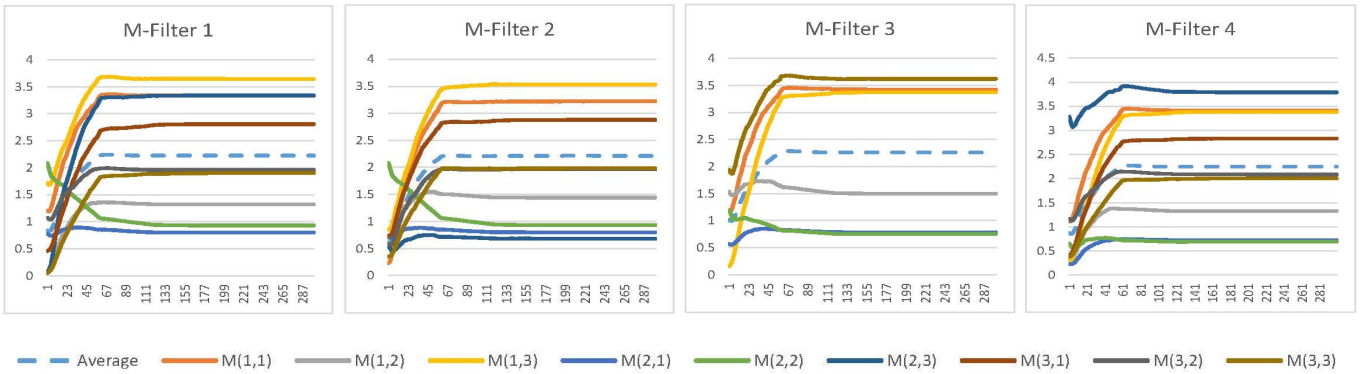


Fig. 11. Curves of elements in M-Filter 1 (M_1'), M-Filter 2 (M_2'), M-Filter 3 (M_3'), and M-Filter 4 (M_4') [in Fig. 2(a) and (3.10)] on the CIFAR experiment in the training process. The values of the nine elements in each M-Filter are learned similar to their averages (dotted lines), which validates that the special MCNs-1 with a single-average element in each M_j' matrix is reasonable and compact without large performance loss.

in Table III. The LBCNN utilized on SVHN has 80 convolutional layers (40 LBCNN modules), 512 LBC filters, 16 output channels, and 512 hidden units in the fully connected layer. Compared with LBCNN, MCNs obtain a better performance with 1.2% improvement. Note that we only use a subset of the whole SVHN to train our MCNs, while other models use the whole set (including the additional images) to do their training.

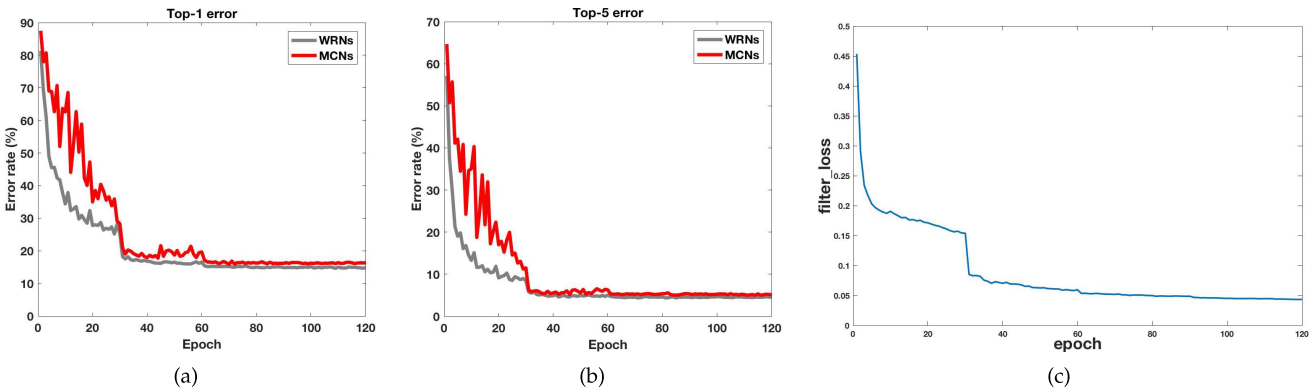
3) CIFAR-10/100: The models and training parameters of MCNs used on CIFAR-10 and CIFAR-100 data sets are the same. The architecture of MCNs has 34 layers with the basic block(c) in Fig. 4, 64-64-128-256 network stage, and 512 hidden units in the fully connected layer. The accuracy of MCNs on CIFAR-10 and CIFAR-100 reaches 95.39% and 78.13%, respectively. Table III shows that MCNs obtain the best performance compared with other state-of-the-art binary methods and other CNNs on both CIFAR-10 and CIFAR-100. In comparison with ResNet-101, MCNs also achieve better performances, which further validate the effectiveness of our model. Fig. 9 shows the training and testing curves of MCNs and U-MCNs on CIFAR-10. Besides, we have also tested MCNs based on VGG-16. And it achieves 93.42% on CIFAR-10, which is even comparable to the full-precision VGG-16 (93.68%) and further validates the effectiveness of MCNs.

4) Discussion on MCNs versus MCNs-1: We have compared MCNs-1 with MCNs on four data sets mentioned earlier, which shows that MCNs-1 achieves a similar performance as MCNs. For example, MCNs-1 and MCNs, respectively, achieve 95.47% and 95.39% on CIFAR-10 and 77.96% and 78.13% on CIFAR-100. The reason should be that the gradient change caused by the binarization is not very large. Our architecture also uses a set of binarized filters to recover unbiarzied ones, which prevents the performance loss.

We also show the curves of M-Filters (MCNs) and their averages (MCNs-1) during training in Fig. 11, with the increasing epochs both of them converge. In addition, the averages are very similar, which can actually be replaced by a single factor (e.g., the whole average) in our modulation process, and thus, furthermore, simplify MCNs. By this operation, the memory usage of M-filter in MCNs-1 is further reduced by nine times with which in ordinary MCN. Experiments show that the performance of reconstructed filters in MCNs-1 does not decrease significantly.

E. Experiments on the ImageNet Data Set Challenge

To further show the effectiveness of the proposed MCNs method, we evaluate it on the 100-class ImageNet and full ImageNet [43] data sets for comparison. In the two experiments, we train a 34-layer MCN with the stage of 32-64-128-256. The corresponding WRNs are used as baselines. The testing performance of MCNs is reported after 120 epochs of training. The learning rate is initialized to 0.1 and decreases to 0.1 $\times$ per 30 epochs.



(a) (b) (c)

Fig. 12. (a) and (b) are testing error curves and (c) is filter loss curve on the 100-class ImageNet experiment.

TABLE IV
CLASSIFICATION ACCURACY (%) ON 100-CLASS IMAGENET

Method	**MCNs**	LBCNN [10]	WRNs [39]
top-1	83.82	63.24	84.96
top-5	94.8	–	95.64

TABLE V
CLASSIFICATION ACCURACY (%) ON FULL IMAGENET

Method	**MCNs-1**	**MCNs**	**U-MCNs**	XNOR-Net [7]
top-1	66.52	65.12	67.11	51.20
top-5	87.21	87.23	88.09	73.20

1) 100-Class ImageNet: Top-1 and Top-5 errors are evaluated and the results are shown in Table IV. The testing error curves are shown in Fig. 12, where we can see that both have similar convergence rates after 30 epochs. Meanwhile, the best result of LBCNN [10] on 100-ImageNet is presented to compare with ours. The LBCNN has 48 convolutional layers (24 LBC modules), 512 LBC filters, 512 output channels, 0.9 sparsity, and 4096 hidden units in the fully connected layer. It is observed that our MCNs gain an advantage over LBCNN by 20.58% in accuracy. Compared with WRNs, our MCNs have only a little performance drop with the same architecture. The unstable beginning shown in Fig. 12(a) and (b) should be due to the large filter loss that causes a large variation after binarization. As shown in Fig. 12(c), the filter loss at the beginning is large. However, MCNs show stable performance after about 35 epochs, which reveals that the unstable beginning is not affected by the optimization algorithm.

2) Full ImageNet: On the full ImageNet, we train a 34-layer MCN with a width of 32-64-128-256, which is the same as the network on the 100-class ImageNet. Top-1 and Top-5 accuracies are evaluated and the results are shown in Table V. Compared with the full-precision unbinarized MCNs (67.11%), our MCNs achieve the top-1 accuracy of 65.12%, which further validates the performance of MCNs. Surprisingly, MCNs-1 achieves a similar performance as MCNs on the large-scale problem, which leads our schemes to be more suitable for practical applications.

We also compared the performance on full ImageNet with SqueezeNet and MobileNet, when their model sizes are comparable in terms of floating-point parameters. Our MCNs (65.12%) outperform SqueezeNet (60.12%) and MobileNet (57.2%) on the full ImageNet experiment (the MCNs use the same setting as that in experiments of 100-class ImageNet). We note that SqueezeNet and MobileNet compress deep models by reducing parameter numbers, while binarization (used in MCNs) is a different way to compress the model. In principle, our MCNs can be used on any full-precision CNNs, which have been tested on conventional CNNs, ResNets, and WRNs. We believe MCNs can also work with SqueezeNet and MobileNet, which is worth further exploring.

F. Experiments on Object Detection

Object detection is one of the most fundamental problems in computer vision, which aims to detect all instances of objects from a known class, such as people, cars, or faces in an image. It has various real-world applications, ranging from robotics and autonomous car to video surveillance and image retrieval. It is very challenging due to the severe scale variation, viewpoint change, intraclass variation, shape variation, occlusion of an object, and background clutters. Girshick *et al.* [49] proposed a simple and scalable detection algorithm. The innovation of R-CNN lies in the feature extraction of candidate regions through the deep network after the completion of a selective search. Owing to this powerful network, R-CNN increased the detection rate of PASCAL VOC from 35.1% to 53.7%. However, RCNN suffers from the slow speed of training and testing. To remedy this situation, Ren *et al.* [1] introduced a region proposal network (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. With a simple alternating optimization, RPN and Fast R-CNN can be trained to share convolutional features. Faster R-CNN (FRCNN) unifies region proposal, feature extraction, classification, and rect refine into a deep network framework, which greatly improved the detection speed while ensuring the performance. Thus, it has become a classical algorithm in the object detection field. Backbone networks have a significant effect on the performance of FRCNN because a good framework for feature extraction is crucial for object detection. We incorporate our MCNs-1 into two-stage FRCNN [1] as backbone networks. MCNs-1 FRCNN achieves a faster speed on object detection than the traditional FRCNN with a reasonable performance loss. Xu *et al.* [50] proposed an amplitude suppression and direction activation for the FRCNN framework to compress DCNNs for highly efficient

TABLE VI

PARAMETERS EVALUATION OF MCNS-1 FRCNN ON OBJECT DETECTION. TEST MAP ON PASCALVOC 2007 DATA SET. TRAINING METHOD IS VOC2007 ONLY AND VOC2007+2012

Dataset	$\mu = 1$	$\mu = 2$	$\mu = 5$	$\mu = 10$
VOC2007	0.474	0.521	**0.558**	0.486
VOC2007+2012	0.563	0.625	**0.661**	0.611

performance. The shared amplitudes between the full precision and the binary kernels can be significantly suppressed through a simple but effective loss, which is incorporated into the existing FRCNN detector. ResNet-18 [34] is used as the backbone networks in FRCNN that are first pretrained on the ILSVRC CLS-LOC data set [51]. We further show that MCNs-1 can benefit the fast object detection task.

1) Data Sets and Implementation Details: We evaluated our method on the two most widely applied detection data sets: PascalVoc and MS COCO. PascalVOC 2007 data set consists of about 5k train/val images and 5k test images over 20 object categories. We also provide results by training on PascalVOC 2007+2012 train/val and testing on PascalVOC 2007 test. More experiments are deployed on MS COCO 2014 [52], which consists of 240k train/val images, 5k minival images, and 40k test-dev images over 80 object categories.

We train FRCNN using the approximate joint training method with an effective mini-batch size of 4. For anchors, we use four scales with box areas of 64^2, 128^2, 256^2, and 512^2 pixels, and three aspect ratios of 1:1, 1:2, and 2:1. By doing so, 256 anchors are randomly sampled in an image to compute the loss function of the RPN. We rescale the images such that their shorter side is 600 pixels. A learning rate of 0.004 for 12.5k mini-batches, and 0.0004 for the next 5k mini-batches, a momentum of 0.9, and a weight decay of 0.0005 are given.

2) Parameters Evaluation: In classification, the loss function of MCN is made up of three parts, namely, filter loss, softmax loss, and center loss. When we used MCNs-1 FRCNN for object detection, the loss function changed. We replaced "Softmax Loss" with a loss of detection. The expression of the total loss function is as follows:

$$L = L_M + \mu L_{\text{detection}} \tag{4.24}$$

where L, L_M, and $L_{\text{detection}}$ are loss functions, and μ is a balance parameter that we add between the two loss functions. The best detection results can be obtained on the VOC data sets when $\mu = 5$. The test results of MCNs-1 FRCNN corresponding to various μ on VOC2007 data set are shown in Table VI. In the following experiments, when MCNs-1 FRCNN is used, the parameter μ in the loss function is 5. Fig. 13 shows the accuracy for different object categories for MCNs-1 FRCNN. The loss function coefficients μ are 1, 2, 5, and 10, respectively. As can be seen from Fig. 13, the detection accuracy increases continuously when μ changes from 1 to 5. While for the value of μ from 5 to 10, the detection accuracy of the detection network will decrease.

3) Experimental Results and Analysis:

a) Results on PascalVOC data sets: We compare the performance of our results with full-precision FRCNN. The comparison results for object detection are shown in Table VII. Compared with the full precision model, the gap will be decreased if more training data are used, which provides a promising way to improve the performance of binary detectors.

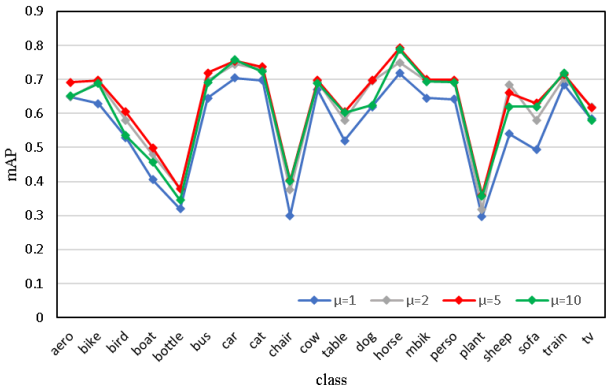


Fig. 13. Results of different classes when μ is taken at different values in MCNs-1 FRCNN.

TABLE VII

TEST MAP ON PASCALVOC 2007 DATA SET IN RESNET-18 BACKBONE. TRAINING METHOD IS VOC2007 ONLY AND VOC2007+2012. "W" AND "A" REFER TO THE WEIGHT AND ACTIVATION BITWIDTH, RESPECTIVELY

Model	W	A	mAP	FPS
VOC2007 Only				
Faster R-CNN-Res18	32	32	67.8	12.26
MCNs-1 FRCNN	1	32	**56.6**	**12.26***
VOC2007+2012				
Faster R-CNN-Res18	32	32	73.2	12.26
MCNs-1 FRCNN	1	32	**66.4**	**12.26***

* Due to hardware constraints, binary acceleration cannot be reflected on the PC, but theoretically it can accelerate 11 times.

TABLE VIII

TEST MAP@.5 AND MAP@[.5, .95] ON MS COCO TEST-DEV IN RESNET-18 BACKBONE. TRAINING METHOD INCLUDES MS COCO TRAIN+VAL

Model	input size	W/A	mAP 0.5	mAP [.5, .95]
YOLOv3-320 [54]	608×608	32/32	59.0	28.2
RetinaNet-50-500 [55]	600×600	32/32	59.0	32.5
CenterNet-Res18 [56]	512×512	32/32	44.9	28.1
Faster R-CNN	1000×600	32/32	44.8	26.0
MCNs-1 FRCNN	1000×600	1/32	**41.5**	**21.4**

b) Results on MS COCO data sets: We compare the performance of our results with other state-of-the-art algorithms, including one-stage fast object detection methods SSD [53], YOLO [54], RetinaNet [55], and CenterNet [56]. The comparison results for object detection are shown in Table VIII.

According to XNOR, MCNs-1 FRCNN can process estimated more images per second (11×), which is faster than most state-of-the-art methods. Together with the 32× memory saving, our detector is very meaningful on mobile and embedded devices. We can also observe that the performance gap of MCNs-1 FRCNN and full-precision counterparts is very small, which further confirms the usefulness of our detector.

c) Results and discussion: Based on the mAP measure in Tables VII and VIII, the results clearly show

that MCNs-1 FRCNN achieves a similar performance to the original Faster RCNN on ResNet-18. We can observe a similar phenomenon that the performance of MCNs-1 FRCNN slightly drops, compared with the full precision FRCNN. Since ResNet is one of the state-of-the-art models, this further confirms that our method is indeed effective on the object detection task. The detection applications of MCNs-1 will be further investigated and developed in our future work.

V. Conclusion

We have developed a new deep learning model, MCNs, which can significantly reduce the storage requirement for computationally limited devices. MCNs (MCNs-1) are implemented by a set of binary filters and the proposed M-Filters. In MCNs, we use M-Filters to build an end-to-end framework and a new architecture to calculate the network model. Both binarized filters and M-Filters are obtained in the same pipeline as in the backpropagation algorithm. The convolution operation is further approximated via the center loss method. MCNs can reduce the storage by a factor of 32, with respect to full-precision models, while achieving a much better performance than state-of-the-art binarized models. Our MCNs based on highly compressed models also achieve a comparable performance to well-known full-precision Resnets or WideResnets. As a general convolutional layer, the M-Filters can also be used in other deep models and different tasks. In future work, we will apply MCNs to audio recognition.

References

[1] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Conf. Workshop Neural Inf. Process. Syst.*, 2015, pp. 1137–1149.

[2] H. Liu, R. Ji, Y. Wu, F. Huang, and B. Zhang, "Cross-modality binary code learning via fusion similarity hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7380–7388.

[3] H. Liu, R. Ji, J. Wang, and C. Shen, "Ordinal constraint binary coding for approximate nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 4, pp. 941–955, Apr. 2019.

[4] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2117–2125.

[5] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 111–118.

[6] Y. Zhou, Q. Ye, Q. Qiu, and J. Jiao, "Oriented response networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 519–528.

[7] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: Imagenet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 525–542.

[8] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to $+1$ or -1," 2016, *arXiv:1602.02830*. [Online]. Available: http://arxiv.org/abs/1602.02830

[9] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3123–3131.

[10] F. Juefei-Xu, V. N. Boddeti, and M. Savvides, "Local binary convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 19–28.

[11] B. Zhang, A. Perina, Z. Li, V. Murino, J. Liu, and R. Ji, "Bounding multiple Gaussians uncertainty with application to object tracking," *Int. J. Comput. Vis.*, vol. 118, no. 3, pp. 364–379, Jul. 2016.

[12] X. Wang *et al.*, "Modulated convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 840–848.

[13] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. Peter Graf, "Pruning filters for efficient convnets," in *Proc. Int. Conf. Learn. Represent.*, 2017.

[14] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2015.

[15] B. Li, B. Wu, J. Su, G. Wang, and L. Lin, "EagleEye: Fast sub-net evaluation for efficient neural network pruning," 2020, *arXiv:2007.02491*. [Online]. Available: http://arxiv.org/abs/2007.02491

[16] A. Zhou, A. Yao, K. Wang, and Y. Chen, "Explicit loss-error-aware quantization for low-bit deep neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9426–9435.

[17] S. Wu, G. Li, F. Chen, and L. Shi, "Training and inference with integers in deep neural networks," in *Proc. Int. Conf. Learn. Represent.*, 2018.

[18] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. Int. Conf. Learn. Represent.*, 2016.

[19] L. Hou, R. Zhang, and J. T. Kwok, "Analysis of quantized models," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–18.

[20] P. Yin, J. Lyu, S. Zhang, S. Osher, Y. Qi, and J. Xin, "Understanding straight-through estimator in training activation quantized neural nets," 2019, *arXiv:1903.05662*. [Online]. Available: http://arxiv.org/abs/1903.05662

[21] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Neural Information Processing Systems*. New York, NY, USA: Curran Associates, 2014.

[22] C. Tai, T. Xiao, Y. Zhang, and X. Wang, "Convolutional neural networks with low-rank regularization," in *Proc. ICLR*, 2015.

[23] Z. Xie, Z. Zhang, X. Zhu, G. Huang, and S. Lin, "Spatially adaptive inference with stochastic feature sampling and interpolation," 2020, *arXiv:2003.08866*. [Online]. Available: http://arxiv.org/abs/2003.08866

[24] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Apr. 2017. [Online]. Available: https://arxiv.org/abs/1704.04861

[25] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," 2017, *arXiv:1707.01083*. [Online]. Available: http://arxiv.org/abs/1707.01083

[26] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," in *Proc. Int. Conf. Learn. Represent.*, 2017.

[27] Z. Wang, W. Zhang, N. Liu, and J. Wang, "Transparent classification with multilayer logical perceptrons and random binarization," in *Proc. AAAI*, 2020, pp. 6331–6339.

[28] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2285–2294.

[29] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proc. Neural Inf. Process. Syst.*, 2016.

[30] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient DNNs," in *Proc. Neural Inf. Process. Syst.*, 2016.

[31] X. Jin, X. Yuan, J. Feng, and S. Yan, "Training skinny deep neural networks with iterative hard thresholding methods," 2016, *arXiv:1607.05423*. [Online]. Available: http://arxiv.org/abs/1607.05423

[32] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556

[33] K. Alex, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[35] R. Li, Y. Wang, F. Liang, H. Qin, J. Yan, and R. Fan, "Fully quantized network for object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2810–2819.

[36] H. Gao *et al.*, "DupNet: Towards very tiny quantized CNN with improved accuracy for face detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2019, pp. 168–177.

[37] H. Peng and S. Chen, "BDNN: Binary convolution neural networks for fast object detection," *Pattern Recognit. Lett.*, vol. 125, pp. 91–97, Jul. 2019.

[38] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2016, pp. 499–515.

[39] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proc. Brit. Mach. Vis. Conf.*, 2016.
[40] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[41] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.
[42] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshop Deep Learn. Unsupervised Feature Learn.*, no. 2, 2011, p. 5.
[43] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[44] A. Banerjee and V. Iyer. (2015). *Cs231n Project Report—Tiny Imagenet Challenge*. [Online]. Available: http://cs231n.stanford.edu/2015/reports.html
[45] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012, *arXiv:1207.0580*. [Online]. Available: http://arxiv.org/abs/1207.0580
[46] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," 2012, *arXiv:1212.5701*. [Online]. Available: http://arxiv.org/abs/1212.5701
[47] I. J. Goodfellow, D. Wardefarley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1319–1327.
[48] M. Lin, Q. Chen, and S. Yan, "Network in network," in *Proc. Int. Conf. Learn. Represent.*, 2014.
[49] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.
[50] S. Xu, Z. Liu, X. Gong, C. Liu, M. Mao, and B. Zhang, "Amplitude suppression and direction activation in networks for 1-bit faster R-CNN," in *Proc. 4th Int. Workshop Embedded Mobile Deep Learn.*, New York, NY, USA, 2020, pp. 19–24, doi: 10.1145/3410338.3412340.
[51] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[52] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Proc. ECCV*. Berlin, Germany: Springer, 2014, pp. 740–755.
[53] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. ECCV*. Berlin, Germany: Springer, 2016, pp. 21–37.
[54] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*. [Online]. Available: http://arxiv.org/abs/1804.02767
[55] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2980–2988.
[56] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," 2019, *arXiv:1904.07850*. [Online]. Available: http://arxiv.org/abs/1904.07850

Baochang Zhang received the B.S., M.S., and Ph.D. degrees in computer science from the Harbin Institute of the Technology, Harbin, China, in 1999, 2001, and 2006, respectively.

From 2006 to 2008, he was a Research Fellow with The Chinese University of Hong Kong, Hong Kong, and Griffith University, Brisbane, QLD, Australia. He was a Senior Post-Doctoral Fellow with the Italy Institute of Technology, Genoa, Italy, from 2014 to 2015. He is currently a Professor with Beihang University, Beijing, China. His current research interests include deep learning, pattern recognition, object recognition and tracking, and wavelets.

Runqi Wang received the B.S. degree in automation from the China University of Mining and Technology, Xuzhou, China, in 2018. He is currently pursuing the Ph.D. degree with the School of Automation Science and Electrical Engineering, Beihang University.

His research interests include computer vision and machine learning.

Xiaodi Wang received the B.S. degree in automation from the Dalian University of Technology, Dalian, China, in 2016. She is currently pursuing the Ph.D. degree with the School of Automation Science and Electrical Engineering, Beihang University.

She was a member of Intelligent Image Analysis and Understanding Laboratory, Dalian University of Technology, led by Prof. Huchuan Lu during her period. She has authored or coauthored four articles in top journals and conferences. Her research interests include computer vision and machine learning.

Jungong Han is currently a tenured Senior Lecturer (Associate Professor) with the Data Science Institute, Lancaster University, U.K. He has been continuously conducting research in the fields of video analysis, computer vision, and machine learning in the past 15 years. He has authored or coauthored over 150 articles in leading journals and prestigious conferences.

Dr. Han is a member of the Editorial Board of several international journals, such as *Neurocomputing* (Elsevier), *Multimedia Tools and Applications* (Springer), and the *IET Computer Vision*. He has been a Guest Editor (leading) for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS and the IEEE TRANSACTIONS ON CYBERNETICS. His one of the first-authored articles has been cited nearly 1000 times.

Rongrong Ji is currently a Professor and the Director of the Intelligent Multimedia Technology Laboratory, Xiamen University, Xiamen, China, where he is currently the Dean Assistant with the School of Information Science and Engineering. His work mainly focuses on innovative technologies for multimedia signal processing, computer vision, and pattern recognition, with over 100 articles published in international journals and conferences.

Dr. Ji is a Member of the Association for Computing Machinery (ACM). He also serves as a program committee member for several tier international conferences. He was a recipient of the ACM Multimedia Best Paper Award and the Best Thesis Award at the Harbin Institute of Technology. He serves as an Associate/Guest Editor for international journals and magazines, such as *Neurocomputing*, *Signal Processing*, *Multimedia Tools and applications*, *IEEE Multimedia Magazine*, and *Multimedia Systems*.